

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-270431

(43)Date of publication of application : 25.09.1992

(51)Int.Cl.

G06F 12/08

G06F 12/08

(21)Application number : 03-085843

(71)Applicant : DIGITAL EQUIP CORP <DEC>

(22)Date of filing : 27.03.1991

(72)Inventor : JOUPPI NORMAN P  
EUSTACE ALAN

(30)Priority

Priority number : 90 499958

Priority date : 27.03.1990

Priority country : US

90 500062

27.03.1990

US

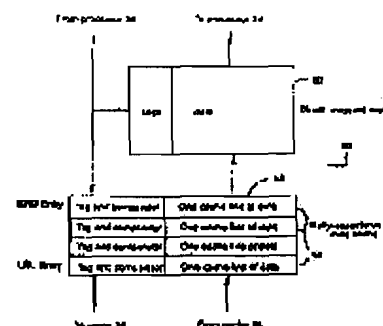
## (54) MEMORY SYSTEM FOR DATA PROCESSOR

(57)Abstract:

PURPOSE: To reduce the miss penalty by improving the performance of a cache memory.

CONSTITUTION: A small complete extension miss cache 42 is provided between a cache 20 of a first level and a cache 26 of a second level to utilize miss caching.

Though a miss penalty in many cycles is given without the miss cache 42, a miss in the cache 20 hit in the miss cache 42 has a miss penalty only in one cycle. A improved method of miss caching where the small complete extension miss cache is victim caching. A stream buffer is started with a cache miss address to prefetch a cache line. It is useful to remove a cache miss of a large capacity.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision]

of rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平4-270431

(43)公開日 平成4年(1992)9月25日

(51)Int.Cl.<sup>4</sup>  
G 0 6 F 12/08

識別記号 庁内整理番号  
F 7232-5B  
3 1 0 7232-5B

F I

技術表示箇所

審査請求 有 請求項の数33(全 19 頁)

(21)出願番号 特願平3-85843

(22)出願日 平成3年(1991)3月27日

(31)優先権主張番号 07/499958

(32)優先日 1990年3月27日

(33)優先権主張国 米国 (U S)

(31)優先権主張番号 07/500062

(32)優先日 1990年3月27日

(33)優先権主張国 米国 (U S)

(71)出願人 590002873

デジタル イクイブメント コーポレイ  
ション

アメリカ合衆国 マサチューセッツ州  
01754メイナード メイン ストリート  
146

(72)発明者 ノーマン ビー ジョウビイ

アメリカ合衆国 カリフォルニア州  
94306 パロ アルト コロラド アベニ  
ュー 617

(74)代理人 弁理士 杉村 暁秀 (外5名)

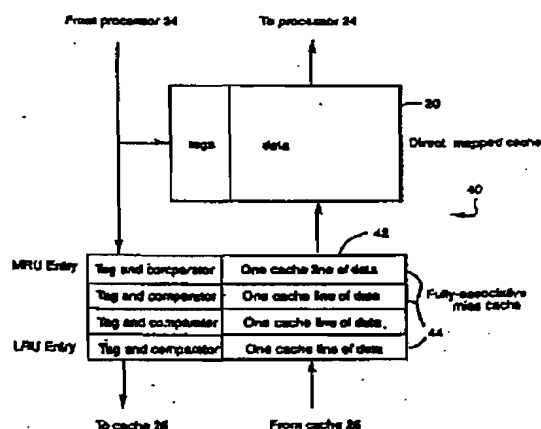
最終頁に続く

(54)【発明の名称】 データ 処理装置のメモリ システム

(57)【要約】 (修正有)

【目的】 キャッシュメモリのパフォーマンスを向上し  
て、ミスペナルティを減少させる。

【構成】 第1レベルのキャッシュ20と第2レベルのキ  
ャッシュ26との間に小さい完全付属ミス キャッシュ  
42を設けてミス キャッシュすることを利用する。ミス  
キャッシュ42がないならば多数サイクル ミス  
ペナルティを持つのに反して、ミス キャッシュ42中  
でヒットするキャッシュ20内のミスは、僅かに1サイ  
クル ミス ペナルティしか持たない。小さい完全付属  
ミス キャッシュを搭載したミス キャッシングの改良  
型がピクティム キャッシングである。ストリーム パ  
ッファはキャッシュ ミス アドレスでスタートして、  
キャッシュ ラインをプリフェッチする。これは容量の  
大きいキャッシュ ミスを除去するのに有用である。



## 【特許請求の範囲】

【請求項1】第1キャッシュメモリ(18, 20)と、第2メモリ(26)と、これらの第1キャッシュメモリと第2メモリの間に接続されたストリームバッファ(62)とを有し、さらに前記第1キャッシュメモリ、第2メモリ及びストリームバッファに接続されている手段(16)を有し、情報をアドレスする第1キャッシュメモリ内にミスが生じたときは、該手段は第1キャッシュメモリ及びストリームバッファに情報をアドレスし、かつ情報を供給し、該情報のアドレス及び供給手段は、前記第1キャッシュメモリ内でミスされた情報を供給し、かつこのミスされた情報に対するアドレスに後続する少くとも1つのアドレス内の情報を前記ストリームバッファに供給する手段を含んでなるデータ処理装置のメモリシステム。

【請求項2】前記第1キャッシュメモリがインストラクションキャッシュ(18)を含む請求項1記載のメモリシステム。

【請求項3】前記第1キャッシュメモリがさらにデータキャッシュ(20)を含む請求項1記載のメモリシステム。

【請求項4】前記情報のアドレス及び供給手段は、前記第1キャッシュメモリとストリームバッファとに同時にアドレス情報を供給する手段を含んでなる請求項1記載のメモリシステム。

【請求項5】前記情報のアドレス及び供給手段は、前記第1キャッシュメモリとストリームバッファとに同時に対応情報を供給する手段を含んでなる請求項1記載のメモリシステム。

【請求項6】第1キャッシュメモリと第2メモリの間に接続された付加的な複数個のストリームバッファを有し、前記情報アドレス及び供給手段は、第1キャッシュメモリ内でミスされたデータを供給し、このミスされたデータのアドレスの後続の少くとも1つのアドレス内のデータを前記付加的な複数のストリームバッファに供給する手段を有してなる請求項1記載のメモリシステム。

【請求項7】情報のアドレス及び供給手段が、ストリームバッファの1つ及び第2メモリよりミスされた情報を供給する手段を含む請求項1記載のメモリシステム。

【請求項8】前記第1キャッシュメモリを直接マップ・キャッシュメモリとする請求項1記載のメモリシステム。

【請求項9】第1キャッシュメモリ(18, 20)、第2メモリ(26)、及び第1キャッシュメモリより小容量で、第1キャッシュメモリと第2メモリの間に接続された付属のミスキャッシュ(42, 52)とを有し、さらに前記第1キャッシュメモリ、第2メモリ、及び付属のミスキャッシュに接続され、アドレスされた情報に

対し第1キャッシュメモリ内にミスが生じたときは、この第1キャッシュメモリ及び付属のミスキャッシュに情報をアドレスし、情報を供給する手段(16)を具えてなるメモリシステム。

【請求項10】第1キャッシュメモリがインストラクションキャッシュを有する請求項9記載のメモリシステム。

【請求項11】第1キャッシュメモリがさらにデータキャッシュを有する請求項10記載のメモリシステム。

【請求項12】前記情報のアドレス及び供給手段は、第1キャッシュメモリ及び付属のミスキャッシュに同時に情報のアドレスを供給する手段を含んでなる請求項9記載のメモリシステム。

【請求項13】前記情報のアドレス及び供給手段は、対応の情報を前記第1キャッシュメモリ及び付属のミスキャッシュに供給する手段を含んでなる請求項10記載のメモリシステム。

【請求項14】前記情報のアドレス及び供給手段は、前記付属のミスキャッシュの1つ及び前記第2メモリより第1キャッシュメモリへ情報を供給する手段を含んでなる請求項10記載のメモリシステム。

【請求項15】前記第1キャッシュメモリが直接マップキャッシュメモリであり、情報のアドレス及び供給手段は、前記付属のミスキャッシュの1つより情報を移動された直接マップ第1キャッシュメモリ及び第2メモリより前記の付属ミスキャッシュに情報を供給する手段を含む請求項14記載のメモリシステム。

【請求項16】情報のアドレス及び供給手段が、バリエイエラーの際、正しい情報を、前記付属ミスキャッシュより直接マップ第1キャッシュメモリに供給する手段を有する請求項15記載のメモリシステム。

【請求項17】インストラクションの一部分を付加的に含んでいるデータを前記情報が有しており、メモリシステムは、前記直接マップ第1キャッシュメモリと第2メモリの間に接続されたストリームバッファを付加的に有しており、前記情報のアドレス及び供給手段は、前記直接マップ第1キャッシュメモリ内でミスされたインストラクションを供給し、かつこのミスされたインストラクションのアドレスに後続する少くとも1つのアドレスにおける情報を、該ストリームバッファに供給する手段を有してなる請求項15記載のメモリシステム。

【請求項18】メモリシステムがさらに、前記直接マップ第1キャッシュメモリと第2メモリの間に接続された複数個のストリームバッファを有し、前記情報のアドレス及び供給手段は、前記直接マップ第1キャッシュメモリでミスされたデータを供給し、ミスされたデータのアドレスに後続する少くとも1つのアドレスにおけるデータを前記付加的な複数個のストリームバ

ッファに供給する手段を有してなる請求項17記載のメモリシステム。

【請求項19】第1キャッシュメモリをアドレスし、このアドレスにตอบสนองして第1キャッシュメモリ内にミスが生じたか否かを決定し、第1キャッシュメモリ内にミスが生じたときは、第2メモリより第1キャッシュメモリ及びミスキャッシュに情報を供給することを特徴とするメモリアクセス方法。

【請求項20】第1キャッシュメモリと、ミスキャッシュとに同時に情報アドレスを供給する請求項19記載のメモリアクセス方法。

【請求項21】第1キャッシュメモリにミスが生じたときは、第2メモリより第1キャッシュメモリ及びミスキャッシュに対応の情報を供給する請求項19記載のメモリアクセス方法。

【請求項22】第1キャッシュメモリにミスが生じたときは、さらにミスキャッシュをアドレスし、ミスキャッシュ内にミスが存しないときはこれにตอบสนองしてミスキャッシュより情報を供給し、第1キャッシュメモリ及びミスキャッシュにミスが生じたときは、第2メモリより第1キャッシュメモリ及びミスキャッシュに情報を供給する請求項19記載のメモリアクセス方法。

【請求項23】ミスキャッシュの1つ並びに第2メモリより移転した第1キャッシュメモリよりの情報をミスキャッシュに供給する請求項19記載のメモリアクセス方法。

【請求項24】パリティエラーを検出し、パリティエラーの存する場合に、ミスキャッシュより第1キャッシュメモリに対応の情報を供給するステップを含んでなる請求項23記載のメモリアクセス方法。

【請求項25】データを有する情報が付加的にインストラクションの一部を含み、さらに本方法は、第1キャッシュメモリ内でミスされたインストラクションを第1キャッシュメモリに供給し、ミスされたインストラクションに対するアドレスに後続する少くとも1つのアドレス内の情報を供給するステップを有してなる請求項23記載のメモリアクセス方法。

【請求項26】第1キャッシュメモリ内でミスされたデータを第1キャッシュメモリに供給し、ミスされたデータに対するアドレスに後続する少くとも1つのアドレス内のデータを付加的な複数個のストリームバッファに供給するステップを含んでなる請求項25記載のメモリアクセス方法。

【請求項27】第1キャッシュメモリをアドレスし、このアドレスに応じてミスが発生したか否かを決定し、第1キャッシュメモリにミスが発生した場合は、第2メモリより第1キャッシュメモリ及びストリームバッファに情報を供給するステップを有するメモリアクセス方法。

【請求項28】第1キャッシュメモリ及びストリームバッファに同時に情報アドレスを供給する請求項27記載のメモリアクセス方法。

【請求項29】第1キャッシュメモリにミスが生じたときは、第2メモリより第1キャッシュメモリ及びストリームバッファに対応の情報を供給する請求項27記載のメモリアクセス方法。

【請求項30】第1キャッシュメモリ内でミスされた情報を第1キャッシュメモリに供給し、かつこのミスされた情報及びミスされた情報のアドレスに後続する少くとも1つのアドレスの情報をストリームバッファに供給するステップを含んでなる請求項29記載のメモリアクセス方法。

【請求項31】ミスされた情報がインストラクションを含む請求項30記載のメモリアクセス方法。

【請求項32】第1キャッシュメモリ内でミスされたデータを第1キャッシュメモリに供給し、ミスされたデータに対するアドレスに後続する少くとも1つのアドレス内のデータを付加的な複数個のストリームバッファに供給するステップを有する請求項31記載のメモリアクセス方法。

【請求項33】第1キャッシュメモリ内にミスが発生するとストリームバッファをアドレスし、このアドレスにตอบสนองしてストリームバッファ内にミスが存しないときはストリームバッファより情報を供給し、第1キャッシュメモリ及びストリームバッファより情報を供給し、第1キャッシュメモリ及びストリームバッファ内にミスが生じたときは前記情報を第2メモリより第1キャッシュメモリ及びストリームバッファに供給するステップを含む請求項27記載のメモリアクセス方法。

【発明の詳細な説明】

【0001】

【技術分野】本発明は、データ処理におけるメモリハイパーキーの性能（パフォーマンス）を向上するシステム及び方法に関する。とくに本発明はキャッシュ（CACHE）メモリのシステムパフォーマンス（性能）を向上するシステム及び方法に関する。さらに本発明は、キャッシュメモリへのアクセスのミスペナルティ（性能低下）を大幅に減少させるシステム及び方法に関する。

【0002】

【従来の技術】キャッシュ（緩衝記憶）機能は、最近のプロセッサの動作機能に極めて重要な効果をもっているのですます重要となってきた。表1は、キャッシュミス時間と、マシンのパフォーマンスのミスの影響を示すものである。最近の10年間に、主メモリのアクセス時間よりもサイクルタイムが遙に急速に減少している。インストラクション当りのマシンサイクルの平均数も顕著に減少しており、とくにコンプレックス（総

5

合) セット コンピュータ (CISC) マシンよりレジュースト (減少) インストラクション セット コンピュータ (RISC) マシンへの転移 (transition) が含まれる場合、これが著しい。これら2つの影響は相乗的であり、ミス動作 コストを極めて増大させる結果を招来する。例えば、デジタル イクイップメント コーポレイション VAX 11/780 のキャッシュ ミスは、平均インストラクション エグゼキューション (実行) の60%を要するのみである。従ってインストラクションがキャッシュ ミスを有していても、そのパフォーマンスは60%のスロー ダウンを示すのみである。しかしデジタル イクイップメント コーポレイション WRL の如き RISC マシンがミスを有すると、そのコストはほぼ10倍のインストラクション タイムとなる。とくにメモリ アクセス タイムとマシン サイクル タイムの比が\*

Machine	cycles per instr	cycle time (ns)	mean time (ns)	miss cost (cycles)	miss cost (in)
VAX 11/780	10.0	200	1200	6	
WRL Titan	1.4	45	540	12	
?	0.5	4	280	70	1.

【0004】

【発明の開示】本発明は、一般に第1キャッシュ メモリと第2メモリを有するメモリ システムに関する。これらの第1キャッシュ メモリと第2メモリの間にミス キャッシュを接続する。ミス キャッシュは、第1キャッシュ メモリよりも小さなメモリ容量を有する。情報のアドレスを行うこれらの第1キャッシュ メモリ、第2メモリ及びミス キャッシュにある特定の手段を接続し、第1キャッシュメモリ内にアドレス情報のミスが発生した際、情報をアドレスし、第1キャッシュ メモリ及びミス キャッシュへの情報の供給を行う。

【0005】本発明によるメモリのアクセス方法は、第1キャッシュ メモリをアドレスし、このアドレスに回答する第1キャッシュ メモリ内にミスが発生するか否かを確かめる。第1キャッシュ メモリ内にミスが発生したときは、第2メモリより第1キャッシュ メモリへ、並びにミス キャッシュに情報が供給される。

【0006】本発明はさらに、第1キャッシュ メモリ、第2メモリ並びに付属のミス キャッシュで、第1キャッシュ メモリよりも小容量を有し、第1キャッシュ メモリと第2メモリ間に接続されているミス キャッシュを有し、さらにこれらの第1キャッシュ メモリ、第2メモリ及び付属のミス キャッシュに接続されており、第1キャッシュ メモリ内に情報のアドレスのミスが生じたときは、第1キャッシュ メモリ及び前記付属のミス キャッシュに情報を供給する手段を具えたメモリ システムに関する。

【0007】

【実施例】以下図面を参照して本発明を説明する。図1

6

\*増加すると、この傾向は継続する。将来、サイクル当りの2つのインストラクションを遂行する超大形マシンで主メモリへ至るすべての通路のキャッシュ ミスは、100 インストラクション タイム以上のコストを要することとなる。良く知られているキャッシュ デザイン技術を注意深く応用した場合でも、100 インストラクション タイム以上の主メモリ回転待時間 (latency) を有するマシンでは、メモリハイアラキーの固有性能 (パフォーマンス) の半分以上を直に失い易いこととなる。このことから最新のメモリ ハイアラキーのハードウェア及びソフトウェアの双方の研究が極めて重要となってきた。

【0003】

【表1】

は、メモリ システムの本発明に関係ある部分のみを示す。同一のチップ22上に、あるいは最新のパッケージ技術によって製造された単一の高速モジュール上に、CPU (中央処理ユニット) 12、フローティング ポイント ユニット (FPU) 14、メモリ マネージメント ユニット (MMU) 16 (例えばトランスレーション ルック・アサイド バッファあるいはTLB)、及び第1レベル インストラクション及びデータ キャッシュ18及び20を設ける。(以下本明細書においては、セントラル プロセッサ24をシングル チップと称するが、これはチップまたはモジュールを意味する。) このチップ22のサイクル時間は、インストラクションの発出速度 (レート) よりも3~8倍長い。(すなわち、3~8インストラクションによって1つのオフ・チップ クロック サイクルとなる。) これは、サイクル当り多くのインストラクションを発出すること (スーパー カレンダまたは極めて長いインストラクション ロード VLIW) により、極めて速いオン・チップ クロックを設けるか (すなわち、通称スーパー パイプライニング)、あるいは、システムの残部に対するよりもプロセッサ チップ22に対し高速技術を用いる (例えば、GaAs 対 BiCMOS) ことによつて達成できる。

【0008】オン・チップ キャッシュ18及び20の予期されるサイズ (寸法) は、プロセッサ24の製造技術によって異なるが、高速技術は一般に小サイズのオン・チップキャッシュを要することとなる。例えば極めて大きなオン・チップ キャッシュ18及び20がCMOSに用いるに適しているが、GaAsまたはバイポーラによるプロセッサには近接限界 (near term) より小形のキャッシュ18及び

7

20しか使用できない。従ってGaAs及びバイポーラはより急速な動作が可能であるが、これらの小形のキャッシュより高いミスレートによって、GaAsまたはバイポーラマシンと、稠密なCMOSマシン間の実際のシステムパフォーマンス比は、これらのゲート速度の比以下となる。これらすべての場合、第1レベルキャッシュ18及び20は、直接マップされるものとした。これはもっとも早いアクセス時間を得られるからである。オン・チップキャッシュ18及び20のライン寸法は、大体16B～32Bの範囲である。データキャッシュ20は、ライト・スルーまたはライト・バックの何れでも良い。本明細書ではこれらの長・短は論じない。

【0009】第2レベルキャッシュ26は、512KB～16MBの容量とし、超高速の静（スタティック）RAMより構成される。これは、第1レベルキャッシュ18、20と同じ理由で、直接マップされる。この大きさのキャッシュ26のアクセスタイムは16～30ns程度である。これはこのキャッシュに、4～30のインストラクションのアクセスタイムを与える。このキャッシュ26のアクセスタイムに比較して、プロセッサ24の相対速度は、第2レベルキャッシュはパイプライン（制御）されるを要し、これによって充分な帯域幅を得るようにするを要する。例えば第1レベルキャッシュ18、20がライトスルーキャッシュである場合を考える。各6または7インストラクション毎に1つの平均速度でストア（蓄積）が生ずるのが典型的であるため、パイプラインされていない外部キャッシュ26は7インストラクションタイムより大なるアクセスタイムに対してはトラフィックの記憶を取扱うに充分な帯域幅を有していない。これ迄何年もの間、キャッシュメモリは主フレーム内で、パイプライン制御されてきていた、しかしこれはワークステーションの最近の開発によるものである。最近ECL I/O及びレジスタまたは入力及び出力にラッチ28を有するキャッシュチップが開発されている。これらはパイプラインされたキャッシュとして理想的なものである。第2レベルキャッシュ内のパイプライン数は2～3として良く、この数は、プロセッサチップ22よりキャッシュチップに至るパイプ段28、及びキャッシュチップよりプロセッサ24に戻るパイプ段がフル（完全）または半（ハーフ）パイプ段であるかによって定まる。

【0010】この速度（例えばMIP当り数メガバイト）のプロセッサ24に充分なメモリを与えるためには、主メモリ34は512MB～4GBの範囲であるを要する。これは約1000個のドラムを含む16Mbのドラム（DRAM）を使用することをも意味する。主メモリシステム34は第2レベルキャッシュのアクセスに比し、大体約10倍の長いアクセスタイムを要する。かかる主メモリ34へのアクセスタイムは、多くのカードに分散している数千のドラム（DRAM）中のアドレス及びデータ信号を抽出（ファンアウト）するに要する時間がその主要部を占る。この

8

ためより高速なドラムが出現しても、主メモリへのアクセスタイムは大体同じ程度に留まる。主メモリへのアクセスタイムが大であることは、第2レベルキャッシュのラインサイズを128または256Bとするを要することとなる。例えば16Bのみの場合は、320ns後に再回（リターン）することとなる。これは50MB/secのバスのバンド幅となる。このバスバンド幅の10MIPプロセッサは、1つのメモリ位置より他のメモリ位置へコピーをするとき、バスバンド幅の制約を受けるので、100～1000のMIPプロセッサを使用してもその能力の追加は僅かである。これはプロセッサのシステム性能（パフォーマンス）を考える上で極めて重要な点である。

【0011】このベースラインシステム10に關いくつかの観察を行うと次の如くである。第1にこのシステムのメモリハイアラキーは、例えばVAX 11/780のマシンと極めてよく類似しており、ハイアラキーの各レベルのみが1レベルCPU12に向って上に移動する。例えばVAX 11/780の8KBのボードレベルキャッシュはオン・チップ移動をする。早期のVAXモデルの512KB～16MB主メモリはボードレベルキャッシュ26となっていた。780型の主メモリがはじめて入力トランスファータ寸法が大となった（ここでは128～256B対VAXの512ページ）。この方式（システム）の主メモリ34は、早期の780型のディスクサブシステムの寸法と同じで、ページング及びファイルシステムキャッシングの如き同じ機能を行う。

【0012】システム10のメモリハイアラキーパフォーマンスの各段階にパラメータを付したものを図2に示した。このシステムに見積られた実際のパラメータは次の如くである。インストラクションイッシュウレート1,000MIP、ファーストレベルインストラクション4KB、4KBデータキャッシュ16Bライン付、1MB第2レベルキャッシュ128Bライン付。ミスペナルティは、第1レベルに対し、24インストラクションタイムであり、第2レベルに対し320インストラクションタイムである。テストプログラムの特性を表2に示す。これらのベンチマーク（水準）は、現在使用されている多くのトレース（計画）に比し、かなり長いものである。しかしこの作業にはマルチプロセッシングの効率はモデルされていない。

【0013】図2に見られるように、殆どのベンチマークは、第1レベルキャッシュのミスで潜在性能（パフォーマンス）の半分以上を失っている。第2レベルキャッシュミスによっては極く少いパフォーマンスが失われるのみである。これは主として、実行プログラムサイズに比較して第2レベルキャッシュのサイズが大であることによるものである。大量のプログラムによるより長いトレースは、極めて重要な第2レベルキャッシュのミスを生ずる。本説明に用いる試験例は、第2レベルキャッシュ機能の有意性に比し極めて少いので、

第2レベル キャッシュ ミスの詳細は検査しない。

【表2】

program name	dynamic instr.	data refs	total refs.	program type
ccom	31.5M	14.0M	45.5M	C compiler
grr	134.2M	59.2M	193.4M	PC board CAD 1
yacc	51.0M	16.7M	67.7M	Unix utility
mat	99.4M	50.3M	149.7M	PC board CAD 1
linpack	144.8M	40.7M	185.5M	numeric, 100x1
liver	23.6M	7.4M	31.0M	LFK (numeric loops)

【0014】想定されるパラメータは所定範囲の極端のもの（最大パフォーマンス プロセッサに最小サイズの  
10 のキャッシュ）であるため、他の形態はメモリ ハイア  
ラーキーのパフォーマンスにおける対応のロス、こ  
れより少ないものとなる。それにも拘らず、興味ある範囲  
内での任意の形態は、メモリ ハイアラーキーの潜在パ  
フォーマンスのかかなりの比率部分を失う。これはシス  
テムのパフォーマンスの中の最大の部分は、メモリ  
ハイアラーキー パフォーマンスの改良によって得ら  
れ、これはCPU のパフォーマンスをより増加しようと  
する試みによって得られるものではないことを意味す  
る。（後者は例えば、インストラクションの並列発生を  
20 より増加させることにより。）以下本明細書で述べる主  
課題は、低価格で、ベースライン メモリ ハイアラー  
キーのパフォーマンスを改良向上させることである。最  
後に付け加えると、CPU コア（CPU12、EPU14、MMU16、  
第1レベル キャッシュ18及び20を含む）のパフォー  
マンスのコンプロマイズ（妥協）を避けるため、研究す  
べき本技術に必要なすべての付加的ハードウェアはCPU  
コアの外側（すなわち、第1レベル キャッシュ18及び  
20の下側）でなければならぬ。かくすることにより、  
30 これらの付加的ハードウェアはキャッシュ ミスの際に  
のみ関与することとなり、通常のインストラクションの  
実行に絶対に（クリティカルに）必要な通路内には存し  
なくなる。

【0015】キャッシュのミスは4つのカテゴリーに分  
類することができ、これらは次の如くである。

コンフリクト（抵触）、……conflict

コンパルソリ（強制）、……compulsory

キャパシティ（容量）、……capacity

コヒーレンス（凝集）、……coherence

コンフリクト ミスとは、キャッシュが完全に付属的  
（アソシアティブ）であり、少なくとも最長時間未使用LR  
U（リスト リセントリィ ユースト）の入換え（リ  
ブレースメント）を有しているときには起らないミスで  
ある。コンパルソリ ミスとは、キャッシュが第1にデ  
ータの一部の最初のものであるため、すべてのキャッ  
シュ構成（オーガニゼーション）に必然的なミスである。  
キャパシティ ミスとは、キャッシュのサイズが参照間  
のデータを保持するに充分でないときに生ずる。コーレ  
ンス ミスとは、マルチプロセッサ キャッシュ コン  
システンシィ（確実性）の保存を無効にするとき以外に  
50

は生じないミスである。

【0016】直接マップされたキャッシュは、これらが  
付属性（associativity）に欠けているためより多くの  
コンフリクト ミスを有するが、これらに対するアクセ  
スタイム コストを考慮するとき、これらのパフォー  
マンスはセット・アソシアティブ キャッシュよりも良  
い。實際上、究極的通路が時間を要するRAM のアクセス  
である場合に、この直接マップ キャッシュのみが唯一  
のキャッシュ形態となる。コンフリクト ミスは、すべ  
ての直接マップ キャッシュ ミスの20%~40%の間に  
あることが一般である。図3は、試験モデルに対するコ  
ンフリクトによりミスのパーセントを示すものである。  
第1レベル データ キャッシュ ミスの平均で30%  
は、コンフリクトによるものであり、また第1レベル  
インストラクション キャッシュ ミスの20%はコンフ  
リクトによるものである。これらはかなり大きな比率で  
あるため、直接マップ キャッシュ メモリに境界的な  
アクセス通路を付加することなく、或る程度の付加的付  
属性（associativity）を与えることにより“我々の成  
果（ケーキ）を享受し、さらにその恩恵にあずかる（食  
べる）”ことが好適である。

【0017】図4のミス キャッシュ システム40に示  
す如く、チップ上の第1レベル キャッシュ20と第2レ  
ベル キャッシュへのアクセス ポートの間小形のミ  
スキャッシュ42を配置することによって直接マップ キャ  
ッシュ20の対応性を付加することができる。ミス キャ  
ッシュ42は、データの2~8程度のキャッシュ線44を有  
する小形の完全・アソシアティブ キャッシュである。  
ミスが発生すると、データは直接マップ キャッシュに  
戻されるのみでなく、ミス キャッシュ42にも戻され、  
ここにおいて、LRU（最長時間未使用）アイテムを入換  
える。上位キャッシュ20が検算（prove）される毎に、  
ミス キャッシュ42も検算される。上位キャッシュ20内  
にミスが生じ、アドレスがミス キャッシュ42に一致し  
ているときは、直接マップ キャッシュ20は次のサイク  
ルでミス キャッシュ42より再ロードされる。かくする  
と、長期のオフ・チップ ペナルティが、短期のオンサ  
イクル オン・チップ ミスで置換えられる。この配置  
は、クリティカルな通路が悪化されないという要求を満  
足する。その理由は、ミス キャッシュ自体はプロセッ  
サの実行の通常のクリティカル通路に存しないからであ  
る。



【0018】コンフリクト ミスの除去において、種々のミス キャッシュ構成の成功率を図5に示す。第1に着目すべきことは、ミス キャッシュ42によって、インストラクション コンフリクト ミスよりも、より多くのデータ コンフリクト ミスが除かれることである。これは次の如くして説明できる。ほとんどすべての場合にあってはまるように、キャッシュ サイズよりもプロセッサ（手続）のサイズが小である限り、1つのプロセッサ内の各インストラクションは互にコンフリクトを生じないように、インストラクション コンフリクトは広く広がる傾向がある。従ってインストラクション コンフリクト ミスは他のプロセッサが呼出されるときに起り易い。ターゲット プロセッサは、呼出し（calling）プロセッサに関し、すべての個所にマップされるので、極めて多くのオーバーラップを来す。各プロセッサにおいて、60の異なるインストラクションが実行されるとすると、試験される最大寸法のミス キャッシュ42内で、コンフリクトミスは15ライン以上に分離される。換言すると、小さなミス キャッシュ42は、全オーバーラップを収容しきれず、これを使用しようとする前に反復して再ロード（reload）する必要がある。この型式の参照パターンは最悪のミスキャッシュパフォーマンスを示す。

【0019】一方、データ コンフリクトは極めて接近した間隔で生じうる。2つのキャラクタ スtringを比較する場合を考えると、2つのStringの比較点が同じラインにマップするとすると、互に相異なるStringへの交互の参照（リファレンス）は常にキャッシュ内で失われる。この場合2つのエントリーのみのミスキャッシュ42によりこれらのコンフリクト ミスのすべてが除かれる。これは他方のパフォーマンスの極端な場合であることは明らかであり、図5の結果は関連のプログラムに基づくパフォーマンスの範囲を示している。それにも拘わらず4KB データ キャッシュに対して、2つのエントリー44のみのミス キャッシュ42によって、データ キャッシュ コンフリクト ミスの平均25%、または全データ キャッシュ ミス（図6）の13%が除かれる。ミスキャッシュ42を4つのエントリー44に増加させると、38%のコンフリクト ミスが除去しうるか、全体で36%のデータ キャッシュ ミスが除かれる。4エントリー44のミス キャッシュエントリーによる改良は僅かであり、15エントリーの44を設けたときも、データ キャッシュの全体の減少は、25%増加したのみであった。

【0020】データ キャッシュ20のサイズを倍増させることによってミスが32%減少するので、（データ キャッシュ サイズを4Kより8Kに増加させると、この指標値以上となる）、第1レベル キャッシュ20内の各追加ラインは、約0.13%だけミスの数を減少させる。ミス キャッシュ42は、記憶ビット当りにして、データ キャ

ッシュ20よりもより多くの面積を必要とするが、2ライン ミス キャッシュ42内の各ラインは、ミス率において50倍の大きさのマージンの改良を行うので、レイアウトのサイズの差を補って余りがある。

【0021】図5と図3とを対比すると、コンフリクトによるミスのパーセントが高い程、ミス キャッシュ42がより有効にこれらのミスを消去することが判る。例えば、図3では、“net”は、全体のデータ キャッシュ20のミスに対し最高のコンフリクト ミスを示している。同様に“grr”及び“yacc”は、コンフリクト ミスの平均パーセントよりも大きな比率を示している。ミス キャッシュ42はこれらのプログラムに対しても極めて大きな貢献をする。“linpack”と“ccom”はコンフリクト ミスの最小パーセント値を示し、ミス キャッシュ42はすべてのプログラムのこれらのミスのごく数パーセントを取除く。これはプログラム データ コンフリクト ミスに大きな比率を有しているときは、全体の密度の理由で、これらを或る程度粉碎（cluster）すべきことを示している。これはプログラムが、少数のコンフリクト ミス、例えばミス キャッシュより、給付（benefit）される“liver”の如きミスを含むことを妨げない。しかしコンフリクトミスのパーセントが増加するにつれ、ミス キャッシュによるこれらのミスの除去率も増加する。

【0022】直接マップ キャッシュとミス キャッシュを有するシステムについて考える。ミスが生ずると、データは、ミス キャッシュと直接マップ キャッシュの双方にロードされる。ある意味では、このデータの二重操作は、ミス キャッシュ内の記憶蓄積スペースを浪費する。ミス キャッシュ内の二重化されるアイテム数は、1（ミス キャッシュ マップ内のすべてのアイテムが直接マップ キャッシュ内のラインと同じになっている場合）より、エントリーの全数（ミス キャッシュをヒットしない一連のミスが生ずる場合）の範囲に亘る。

【0023】ミス キャッシュ42をより良好に使用するため、図7のキャッシュ システム50内に示す小形の完全付属（アソシエティブ）ミス キャッシュ52に対する別の置換アルゴリズムを使用することができ。ミスにより要求されるデータをミスキャッシュ42内にロードする代りに、54に示す如く、直接マップ キャッシュ20よりビクティム ラインによって完全付属ミス キャッシュ52にロードすることができる。これを“ビクティム キャッシング（犠牲キャッシング）”と称する。ビクティム キャッシングには、データ キャッシュ20にも、ビクティムキャッシュ52にも何れにもデータ ラインは現れない。これは、ビクティムキャッシュ52は、直接マップ キャッシュ20より放出されるアイテムのみがロードされるからである。ビクティム キャッシュ52をヒットするような直接マップ キャッシュ20内のミス

13

の場合には、直接マップ キャッシュ ラインとマッチするビクティム キャッシュ ライン56の内容をスワップ(交換)する。

【0024】参照ストリームによって、ビクティム キャッシュはミス キャッシュに小さなあるいは極めて重大な改良の何れかを行うことができる。この利点はミスキャッシュ内の二重化に対応して定まる。ビクティム キャッシュは常にミスキャッシュの改良を行う。

【0025】1例として、コール サイトとコンフリクトを生ずる内側ループ内の小手続き(プロセジャ)を呼出すインストラクション参照ストリームを考える。このプログラムのエクゼキューション(実行)は、ミス キャッシュ42内の位置数よりも大きなコンフリクト ループに沿ったバス長を有することもある。この場合、ミス

キャッシュは価値を生じないこともある。その理由は、キャッシュ内に常時二重化アイテムがフラッシュされるからである。しかし、これに代えてビクティム キャッシュを使用すると、捕捉しうるバス長はほぼ2倍となる。これは1組のコンフリクト インストラクションが直接マップ キャッシュ20内で生きており、また他の1組がビクティム キャッシュ52内で生きているからである。このループに沿ってエクゼキューションが進行するにつれて、これらのアイテムのトレードが行われる。

【0026】ビクティム キャッシングによって除かれるコンフリクト ミスのパーセントを図8に示した。1つのライン56で構成されているビクティムキャッシュ52は、必ず2つのライン44で構成するを要するミス キャッシュ42と逆に有用でさえある。ミス キャッシュと比較してすべてのベンチマークは改善されている。しかしベンチマークのインストラクション キャッシュ18のパフォーマンス及びデータ キャッシュ20のパフォーマンスは長い連続参照ストリーム(例えば、“ccom”及び“リンバック”)がもっとも改良されている。

【0027】図9は、ビクティム キャッシュによって得られたと思われるミスの全体の減少率を示す。図6と図9との比較より判るように、ビクティム キャッシュ52のパフォーマンスは、2倍のエントリー数を有しているミス キャッシュ42よりも場合によって良好である。例えば“yacc”のデータ キャッシュ20のパフォーマンスを、1・エントリービクティム キャッシュ52及び2・エントリー ミス キャッシュ42で考えて見る。ビクティム キャッシュ52はビクティムを放棄しないので、状況によってビクティム キャッシュはエントリー数が2倍のミス キャッシュよりもミスの数が少なくなる。例えば、新規なデータの評価に多くのキャッシュ ミスが生ずる(例えば、強制(compulsory)ミス)ことを考えると、ミス キャッシュ42とビクティム キャッシュ52の双方をフラッシュ アウトすることが有効である。次で他の新しいラインが参照される場

14

合を考えると、ミス キャッシュ42を有するシステム40と、ビクティム キャッシュ52を有するシステム50との双方にミスが生じている。ラインの古い内容が次に参照されると、ミス キャッシュ42はこのアイテムを保有していないが、ビクティム キャッシュ52はこれを保有している。従ってミス キャッシュ42を有するシステム40は、メモリ ハイアラキーの次位のレベルで2つのミスを生ずるが、ビクティム キャッシュ52を有するシステム50は1つのみのミスを生ずることとなる。

【0028】図10は、種々のサイズの直接マップ データ キャッシュ20をバックアップするときの、1、2、4、15エントリーのビクティム キャッシュのパフォーマンスを示す。一般に小さい方の数の直接マップ キャッシュ20が、ビクティム キャッシュ52の付加によって、より多くの利益を得る。参考のため、各キャッシュ サイズに対するコンフリクト ミスの全パーセントを示してある。ビクティム キャッシュ52のパフォーマンス対直接マップ キャッシュ20のパフォーマンスには2つのファクターが存する。その第1は、直接マップ キャッシュ20のサイズが大となるにつれ、ビクティム キャッシュ52の相対的サイズは小となることである。直接マップ キャッシュ20が大となり、ライン サイズを小(16B)としている為、ビクティム キャッシュにより容易に除きうる稠密なマッピングのコンフリクトの可能性が減少する。第2には1KBより32KBへ向ってコンフリクト ミスは値が減少する。既述の如く、コンフリクトミスのパーセントが減少するにつれ、ビクティム キャッシュ52によって除かれるミスのパーセントも減少する。しかし極めて大きなキャッシュに対しては、コンフリクト ミスのパーセントが増加するため、第1の効果がより大きな影響を及ぼし、ビクティム キャッシュ52のパフォーマンスは僅かしか増加しない。

【0029】図11は、種々のライン サイズの4KB直接マップ データ キャッシュ20に対するビクティム キャッシュ52のパフォーマンスを示す。予期したように、このレベルでライン サイズが増加するにつれて、コンフリクト ミスの数も増加する。コンフリクト ミスのパーセントが増加すると、ビクティム キャッシュ52によって除去されるかかるミスのパーセントも増加する。ビクティム キャッシュ52を有するシステム50は、長いライン サイズにおいて、ビクティム キャッシュを設けないものよりも利益を得る。その理由は、ビクティムキャッシュ52は、長いキャッシュ ラインより由来したコンフリクトによるミスを除く助けをするからである。ビクティム キャッシュ52内にデータ記憶のために用いられる面積が一定に保持されるとしても(例えば、ライン サイズが倍となると、エントリー数は半分にカットされる)、ライン サイズが増加したとき、ビクティム キャッシュ52のパフォーマンスは

より改良されるか、少くとも同じ状態を保つ。

【0030】キャッシュのサイズが増加すると、ミスの大きなパーセントは、コンフリクト及びコンパルソリーミスとなり、キャパシティ ミスの比率は減少する。

(キャッシュが全プログラムよりも大きい場合は、コンパルソリー ミスのみが残るので、当然これを除く。) 従ってビクティム キャッシュ52は、第2レベルキャッシュ26にも有効であると期待される。ライン サイズの増加とともに、コンフリクト ミスの数が増加するため、第2レベルキャッシュ26のライン サイズが大となると、ビクティム キャッシュ52の潜在的有用性が増加する傾向にある。第1レベル キャッシュ20の場合と同様に、第2レベル キャッシュ26内でコンフリクト ミスのパーセントが大となるにつれて、ビクティム キャッシュ52によって除かれるコンフリクト ミスのパーセントが大となる。

【0031】ビクティム キャッシュの興味ある一面は、キャッシュのハイアラキーにおけるインクルージョン プロパティ (算入特性) を冒涇 (violate) することである。これは、マルチプロセッサのキャッシュのコンシステンシー (一貫性) のアルゴリズムに悪影響を及ぼすだけでなく、キャッシュ シミュレーションに用いられるアルゴリズムにも影響を及ぼす。例えば、所定のCPU 基準ストリーム上の第2レベル キャッシュ26に対するミスの数は、その頂上 (トップ) にある第1レベル キャッシュ26のサイズ (但し、ライン サイズではない) に無関係である。しかしながらビクティム キャッシュ52は、第1レベルのみでなく、第2レベルにおいてもコンフリクトする多くのラインを含有することができる。このため、第1レベル ビクティム キャッシュ52の使用は、第2レベルにおけるコンフリクト ミスの数も減少させることができる。第2レベル キャッシュ26に対するビクティム キャッシュ52の研究に当り、第1レベル ビクティム キャッシュ52を設ける場合と、設けない場合との両方の形態を考える必要がある。

【0032】メガ バイトの第2レベル キャッシュ26に対するビクティム キャッシュ52の念入りの調査には、数百万のインストラクションのトレースが必要となる。現在我々はより小さいテスト ケースに対するビクティム キャッシュのパフォーマンスのみを有しており、多メガ バイトの第2レベル キャッシュ26に対するビクティム キャッシュのパフォーマンスを得ることについては目下取進め 중이다。

【0033】とくに第1レベルのオン・チップにおけるミス キャッシュ42のその他の重要な用途は、イールド (歩留り) の向上である。すべてのインストラクション及びデータ キャッシュ18及び20のバイトにパリティが維持されており、データ キャッシュ20がライト・スルーである場合、キャッシュ パリティ エラーはミスと

して取扱うことができる。レフィル バスがキャッシュをバイパスするときは、このスキームは、ハードのエラーを有するチップを使用することを許容する。(實際上、バイトのパリティに対しては、1バイト当り、最大で1つの悪いビットがあるとする、キャッシュ内のすべてのビットの1/9 迄が誤っていることもありうる。) 残念ながら、ミス キャッシュ42が存せず、linpack (例えば、saxpy) の内側ループが欠陥のあるライン上にランダムな、構造変数に用いる周波数が欠陥ライン上のものである場合は、システムのパフォーマンスは大幅に低下 (degrade) する、(例えば、あるコード セグメントで係数4より大にデグレードする)。さらに、欠陥の位置によって、ランダムな様相でパフォーマンスのデグレードがチップ毎に変化する。これは、プロジェクトのエンジニアリングの開発における潜在性イールドの強化を制限する。しかしミス キャッシュ42を付加することによって、欠陥を引起すパリティ ミスのペナルティは1サイクルのみとなり、これはオフ・チップ ミスよりもマシンのパフォーマンスに与えるインパクトは遙に小となる。従って欠陥の数が小であり、ミス キャッシュ42で充分取扱えるものであれば、ハードの欠陥のあるチップを生産系で使用することができる。もし、ミス キャッシュ42を、生産上の欠陥のあるシステムのパフォーマンスの改良に使用すべきときは、インストラクション ミス キャッシュあるいは単に1つのエントリーのみを有するミス キャッシュも有用である。

【0034】前に述べたビクティム キャッシュ52はパリティ エラーに起因するミスの修正には有用ではない。これはビクティムがパリティ エラーでコラプス (退化) しており、セーブに値しないからである。しかしビクティム キャッシュ52は次の如くの変化を加えることによってエラー修正にも使用することができる。キャッシュ ミスがパリティ エラーによって生ずる場合には、ビクティムキャッシュ52に入力 (ミス) データをロードし、ビクティムをロードしない。かくすると、通常のミスに対するビクティム キャッシュ52のように動作し、ミス キャッシュ42はパリティ ミスに対し動作する。このような僅かな変形によって、ミス キャッシュ42がエラーのレカバリーに用いられ、ビクティムキャッシュのより良好なパフォーマンスを組合せることができる。

【0035】コンパルソリー (強制) ミスは、如何なるキャッシュ構成でも、一部のデータに最初に参照されるために必要とされるミスである。キャパシティ ミスは、キャッシュの大きさが、参照の間にデータを充分保持するものでないときに生ずるミスである。キャパシティ ミス及びコンパルソリー ミスを減少させる1つの方法は、長いキャッシュ ライン サイズまたはプレフェッチ方法の如きプレフェッチ技術を用いることである。しかしながら、ミスのレートを増加させることなし

に、また転送すべきデータの量を遙に増加させることなしにライン サイズを任意に大とすることはできない。本章では、長いライン及び過剰のプレフェッチに関する従来の問題を減少させ乍ら、キャパシティ ミス及びコンパルソリーミス減ずる技術の研究することとする。

【0036】ライン サイズが長くなると、各種の異なるプログラム及びアクセス パターンに対し固定した転送(トランスファー) サイズを設ける点で不利益を生ずる。プレフェッチ技術は、プログラムの実際のアクセス パターンにより良く適合しているのが興味がある。このことは、インストラクション ストリームまたは、ユニット ストライド アレイ アクセスの如き、長い準連続(quasi-sequential) アクセス パターンのパフォーマンスの改良にとくに重要である。

【0037】3つのプレフェッチ アルゴリズムの詳細な解析が、スミス アラン ジュニア (Smith Alan, J.) により "Cache Memory" として、Computing Surveys 1982, 9, pp473-530 に発表されている。プレフェッチでは、各参照後常にプレフェッチを行う。これは我々のベース システム10では実面的でない。その理由は、単一のレベル・2 キャッシュ レファレンス (参照) を必要とする時間内に数多くのレベル・1 キャッシュ アクセスが行われるからである。これはインストラクション キャッシュ18より、1サイクル当り複数のインストラクションをフェッチし、同時に1サイクル当りデータ キャッシュ20にロードまたは蓄積を行うマシンにおいて特に事実である。プレフェッチ・オン ミス(ミスの際のプレフェッチ) 及びタグド・プレフェッチ(タグ付プレフェッチ) はより有望な技術である。プレフェッチ・オン ミスにおいては、ミスが生ずると、次のラインもプレフェッチする。この技術では、純粋シーケンシャル レファレンス ストリームに対するミスの数を半分にカットする。タグド・プレフェッチは、さらにこれより良好である。この技術においては、各ブロックは付随するタグ ビットを有する。あるブロックがプレフェッチされると、そのタグ ビットは"0" (ゼロ) にセットされる。このブロックが使用される毎にタグ ビットは1にセットされる。このブロックが0より1に転移する毎にその後続のブロックがプレフェッチされる。フェッチが充分急速に行われると、これにより純粋シーケンシャル レファレンス ストリーム内のミスの数は0に減少する。残念乍ら、ベースシステム10では、レーテンシ(潜在性)が極めて大であるためこれは不可能である。図12は、C コンパイラ ベンチマークの遂行中に必要とするプレフェッチ ラインに至る迄の時間(インストラクション 発出迄の時間)を示すものである。当然のことながら、ライン サイズは4インストラクション分であるため、マシンをキャッシュされないストレート・ライン コードで維持するためには、プレフェッチ ラインは、4インストラクション時

間内に受信されなければならない。ベース システムでは、第2・レベル キャッシュ26は、アクセスに多くのサイクルを要し、かつマシンは各サイクル当り実際に数多くのインストラクションを発出するため、タグド プレフェッチのみが、所要のインストラクションを設けるのに、ワン・サイクル・アウト・オブ メニー ヘッド スタートを有する。

【0038】本発明で必要なことは、タグ転移の生ずる前にプレフェッチをスタートさせることである。図13に示したシステム60におけるストリーム バッファ62と称されるメカニズム(機構)によりこれを行うことができる。ストリーム バッファ62は、それぞれタグ66より成る一連のエントリー64、利用可能ビット68及びデータライン70より成る。

【0039】ミスが発生すると、ストリーム バッファ62は、ミス ターゲットより出発して、順次のラインのプレフェッチを開始する。各プレフェッチリクエストが送出されると、このアドレスに対するタグ66がストリーム バッファ62に入力され、ここで利用できるビット68は誤り(false)としてセットされる。プレフェッチデータ70が戻ってくると、そのタグ66と共にエントリー64内に収納され、ここで得られるビット68は真値(true)とされる。ミスによってリクエストされる順次のラインは、キャッシュ20でなくバッファ62に収納される。かくすることによって、必要でないデータによってキャッシュ20が汚染されることが防止される。

【0040】次々のキャッシュへのアクセスにおいても、これらのアドレスをストリーム バッファ62に記憶されている第1アイテムと比較する。この参照において、キャッシュ20ではミスし、ストリーム バッファ62ではヒットしたとすると、キャッシュ20はストリーム バッファ62より単一サイクルで再ロードすることが可能である。これはオフ・チップ ミス ペナルティよりも遙に急速である。ここにおけるストリーム バッファ62は簡単なFIFO キュー(待合せ)と考えられ、キューの初頭のもののみが、タグ比較器72を有しており、ストリーム バッファ62より移動されたエレメントは、如何なるラインをもスキップすることなしに順次厳密に移動させることを要する。この簡単なモデルでは、非順番(non-sequential)ライン ミスは、キューの次の下位に要求されるラインが既に出現していても、ストリーム バッファ62をフラッシュさせ、ミス アドレスより再スタートさせる。以下に、既にフェッチしたラインを順番(シーケンス)外としうるより複雑なストリーム バッファについて述べる。

【0041】ライン64がストリーム バッファ62よりキャッシュ20に移されると、ストリーム バッファ62内のエントリーは1つだけシフトし、新規な後続のアドレスがフェッチされる。次の後位アドレスはインクレメンタ74によって発生される。第2レベル キャッシュ26への

パイプラインインタフェースによって、ストリームバッファ62は第2レベルキャッシュ26の最大帯域幅に充填され、プロセスに当たり、多くのキャッシュラインを同時にフェッチすることができる。例えば、インストラクションキャッシュ18のミスの16Bラインの再充足のレーテンシ（回転待ち時間…latency）が12サイクルと見なす。パイプラインされたメモリインタフェースで、各4サイクル毎に新しいラインリクエストを受入れられるものを考える。4・エンタリーストリームバッファ62は、常時3つのリクエストが待機している状態で、サイクル当たり1つの速度で4Bのインストラクションを与えることができる。従って順次のインストラクション実行中、長いレーテンシキャッシュミスは生じない。これは、同時には1ラインのみしかプレフェッチされない純粋順次参照ストリームにおけるタグ付プレフェッチのパフォーマンスと違っている。ここでは順次のインストラクションは、各3サイクルに1つのインストラクションに等しい帯域幅（すなわち、12サイクルレーテンシ/4インストラクション各ライン当り）でのみ供給される。

【0042】図14は、それぞれ16バイトのラインを有し、4KBインストラクションキャッシュ18をバックする4・エンタリーインストラクションストリームバッファ62と、4KBデータキャッシュ20をバックするデータストリームバッファ62のパフォーマンスを示す。この図は、バッファが最初ミスで開始して、プレフェッチを許されるライン数に基づき、除かれるミスの累計数を示すものである。ほとんどのインストラクションの参照（レファレンス）は、6番目の連続ラインがフェッチされる時まで、純粋な順番アクセスパターンを破り、一方多くのデータ参照パターンはより早く終結する。この例外は、“liver”に対するインストラクション参照及び“linpack”に対するデータ参照である。“liver”は、プログラムの14のループが順次実行され、初めの14ループは一般に他の手続きをコールしないか、あるいは過剰のブランチを行い、連続ミスパターンを破るため、変則である可能性がある。“linpack”のデータ参照（レファレンス）パターンは次の如くして理解できる。ストリームバッファ62は、キャッシュ18または20がミスしたラインを提供するのみの責務を有している。“linpack”の内側のループ（例えば、saxpy）は、マトリックスの1行と他の行との間の内積（inner product）を行う。1つの行の第1の使用により、この行はキャッシュにロードされる。キャッシュの当該の後続のミス（第1行のマッピングコンフリクトを除く）の後、マトリックスの次のラインが構成される。マトリックスは過大であって、オン・チップキャッシュに適さないで、全マトリックスは各反復（iteration）毎にキャッシュを通過する。ストリームバッファ62は第2レベルキャッシュ26によって提供される

最大帯域幅でこれを行いうる。基準ストリームがユニット・ストライドであるか、または各第3ワードまたは最大で他へスキップすることはこれに対し必須の要件である。非・ユニット・ストライド方向にアレイがアクセスされると、（さらに、他のディメンションが、ノン・トライビアルである限り）ここに説明したストリームバッファ62は僅かな利点しか有さなくなる。

【0043】図15は、3つの典型的なストリームバッファにおける帯域要求を示すものである。“ccom”に対する1・ストリームは極めて正規である。（インストラクション中に測定したとき。）平均として、各4.2インストラクション毎に16Bラインをフェッチするを要する。プログラムが短いループに入ると、ストリームバッファ62を参照する間隔が増加し、例えば疑問（else）クローズをスキップするような場合の如く、プログラムが小幅の前方ジャンプを行うようなときはこれは減少する。それにも拘らず、フェッチ周波数は極めて規則正しい。このデータは、例えばデジタルイクイップメントコーポレーションマルチタタCPUまたはMIPS R2000（商品名）の如く、短い機能ユニットレーテンシ（回転待ち時間）を有するマシンに対するものである。従ってインストラクション当りのサイクル数は、キャッシュミスの無いとき、1に極めて近くなる。

【0044】“linpack”及び“ccom”に対するデータストリームバッファの参照のタイミングを図15に示してある。“linpack”に対する新規な16Bラインの参照速度は、各27インストラクション当たり、1である。“linpack”のこの部分は二重精密（double precision）であるため、この作業は、各13.5インストラクション当たり、内側ループの新しい反復（iteration）を行う。これは希望値よりも大である。この“linpack”のバージョンはある程度ルーズであり、各アレイ素子に対するアドレス計算を整数倍し、ループはアンロールされない。ループがアンロールされ、広範な最適化（optimization）が行われるとすると、参照のレート（比率）は増加する。しかし、インストラクション側には存しないデータ側の記憶トラフィックに起因するインストラクションストリームのレートよりも以下でなければならない。“ccom”は興味あるトライモード（三モード）のパフォーマンスを有している。ミスに続いて、後続のラインが使用されるとすると、平均でミス後、僅か5サイクルで必要とされる。ミス後次の2つのラインに対し、平均で各10インストラクション毎に連続するデータライン（16B）が必要とされる。初めの3つのラインがストリームバッファ62の殆どの（82%）利点を生ずる。その後は連続するラインは“linpack”に近いレート、すなわち平均で各24インストラクション毎に必要とされる。

【0045】一般に、バック・アップのメモリが新規ワード（4B）の各サイクルに平均帯域幅でデータを形成

することができると、ストリーム バッファ62は連続参照を維持することが可能となる。これはインストラクション ストリームに対し充分であり、かつ極めて多く巻戻し(アンロール)されたブロックコピーにも充分で、二重プレジジョン ロード及びメモリ(store)を使用する。この帯域幅が得られないときは、インストラクション ストリームバッファの利点は減少し、ブロックコピー及び他の類似のオペレーションも負のインパクトを受ける。しかし各1.5 ~ 2サイクル当り1つの新しいワードを均等化(イコーリング)する帯域幅は多くのデータに対し依然として充分役に立つ。なおこれらの価は帯域幅に対するものであり、図12のプレフェッチ スキムにおいて要求される全レーテンシを達成するよりも遙に容易である。

【0046】前章に述べたストリーム バッファ62は、インストラクション キャッシュ18のミス全体で72%取除くことができた。しかしデータ キャッシュ20のミスは、その25%しか取除くことができない。この理由の1つは、データの参照が、異なるデータ源よりのインタリーブされたデータ ストリームで構成されていることによる。データの参照におけるストリーム バッファ62のパフォーマンスの向上を図るため、マルチ・ウェイストリーム バッファ62のシステム80をシミュレートした。(図16) このシステム80は4個のストリーム バッファ62を並列にして構成されている。何れのストリーム バッファ62をもヒットしないデータキャッシュ20にミスが生ずると、最低頻度(least recently)でヒットされたストリーム バッファ62をクリアし(すなわちLRU置換)、ミスのアドレスにおいてフェッチを開始する。

【0047】図17はマルチ・ウェイ ストリーム バッファ システム80を我々のベンチマーク セットで動作させたときのパフォーマンスを示す。予期したように、インストラクション ストリームによるパフォーマンスは本質的には変らなかった。これはインストラクション ストリームに対しては、より簡単な単一ストリームバッファ システム60で充分間に合うことを示している。しかし、マルチ・ウェイ ストリーム バッファ システム80は、データ側では画期的な性能向上改良を示し、これは6つのプログラムに対しミスの43%を除去することができ、単一ストリーム バッファ システム60の性能のほぼ2倍であった。“liver”のマトリクス動作が最大の改良(減少を7%より60%に変えた)を示したが、すべてのプログラムが或程度の改良を示した。“liver”もそのデータ構造に、ユニット ストライド アクセスを行うことを付記する。

【0048】以上の説明においては、ストリーム バッファ62に対し、単に1つのアドレスコンパレータのみが設けられていた。これは、要求されたラインがストリームバッファ62内であるが、コンパレータ72に対する第1

位置にない場合には、ストリーム バッファ62は参照の際にミスをし、その内容はフラッシュされることを意味する。このスキムに対し、明らかな改良を行う1つは、コンパレータをストリーム バッファ62の各位置に設けることである。かくすると、例えば準連続参照パターンによって、あるキャッシュ ラインがスキップされても、ストリーム バッファ62は、キャッシュ ラインが既にフェッチされている限り依然としてこれに供給を行うことが可能である。

【0049】図18は、3つの比較器(comparators)を持つストリーム バッファの性能を示す。疑似ストリーム バッファ(quasi-stream buffer)は、命令キャッシュミスの76%を除去することができ、これは純粋逐次(purely sequential)ストリーム バッファよりも4%の改善であって、残留ミスの数の14%の減少をもたらす。このことは恐らく、“if”ステートメント中の“then”クローズや“else”クローズのようなコードがスキップしたときに、疑似ストリーム バッファが有用なフェッチを継続できる能力に依るものであろう。シミュレートされたこの変形は3つの比較器を持つので、最大2つのキャッシュ ラインに加えて更に3/4までのキャッシュ ラインを、アラインメントによりどちらかの側で、最大合計16ないし22の命令に対してスキップすることが出来た。このことは、ストリーム バッファがフラッシュされることの起きない(分枝アラインメントによる)逐次ストリーム バッファでは、僅かに0ないし6命令しかスキップしないのに対比される。

【0050】疑似ストリーム バッファの余分の比較器はまた、4方向(four-way)データストリーム バッファの性能をも改善する。結局全体では、4方向疑似ストリーム バッファはすべてのミスの47%を除去することができ、これは純粋逐次4方向ストリーム バッファよりも4%の向上である。

【0051】単一ストリーム バッファへ数個の余分な比較器を設けるために所要のハードウェアの量は小さいものだから、疑似ストリーム バッファは、命令ストリームに対する逐次ストリーム バッファの有益な一般化であるかのように思われる。それは、僅かに2つの比較器を付加することが、逐次ストリーム バッファを疑似ストリーム バッファに転換するのに要求されるだけだからである。しかし、多方向(multi-way)データ疑似ストリーム バッファに対しては、それは有用ではないかも知れない、と云うのは所要の余分な比較器の数が何倍にも大きくなるであろうからである。ソフトウェア探究での興味ある分野として、コンパイラがコードを再編成し、データ レイアウトがストリーム バッファの用途を最大にする能力がある。もし参照規準の逐次性(sequentiality of references)を最適化する技術が成功するならば、ストリーム バッファへの余分な比較器の必要性はさらに小さくなる。

【0052】ストリームバッファの性能を視野に収めるために、このセクションではストリームバッファの性能を以前に文献で検討したプリフェッチ技術に替えることにする。我々の6つのベンチマーク上での、ミスの際のプリフェッチ(prefetchon miss)、タグ付プリフェッチ(tagged prefetch)及び常時プリフェッチ(always prefetch)の性能が表3に示される。このデータは、1命令付与(oneinstruction-issue)の第2レベルキャッシュ回転待ち時間(latency)を伴うこれらのプリフェッチ技術の使用を前提にして、ミスの減少を示している。1命令付与の回転待ち時間は1マシンサイクルより小さいであろうから、また第2レベルキャッシュは典型的に多くのCPUサイクルの回転待ち時間を持つことから、このデータは全く非現実的なものであることに留意されたい。それにも拘らず、これらの数字はこれらのプリフェッチ技術の性能の上限を与えるのである。この検討におけるプリフェッチアルゴリズムの性能は以前の文献に示されたデータとよく一致している。上に引用したスミス(Smith)の論文では、16Bラインと8方向セット結合性(16B lines and 8-way set associativity)を伴う8KB混合キャッシュ上のPDP-11痕跡TRACE(a PDP-11 trace TRACE on a 8K Bmixed cache)に対するミスレート(miss rate)の減少は(混合キャッシュだけしか検討されていないので)ミスの際のプリフェッチに対して27.8%、タグ付プリフェッチに対して50.2%、常時プリフェッチに対して51.8%であることが判っている。

\*【0053】表4では、表4からのプリフェッチの性能を以前に示したストリームバッファの性能と比較している。命令の側では、単純単一ストリームバッファ(a simple single stream buffer)62はミスの際のプリフェッチを広いマージンで出力実行している。このことは、純粋逐次参照標準ストリーム(a purely sequential reference stream)に対してミスの際のプリフェッチはミスの数を因数2で減少させるのみであろうから、驚くには当たらない。単純単一ストリームバッファシステム60も疑似ストリームバッファシステム80も共に、タグ付プリフェッチと殆ど同じように機能している。トラフィックに属する限り、ストリームバッファ62はミスの後にタグ付プリフェッチより多くフェッチするであろうが、しかしタグ転移に際してはフェッチを開始しないだろうから、トラフィック率の比較は今後の興味ある研究課題である。命令ストリーム上のストリームバッファ62の性能は常時プリフェッチより僅かに劣る。このことは、常時プリフェッチの性能が分枝を取らない命令の百分率に近似するから、また命令の減少に際し逐次プリフェッチによるキャッシュミスの上限だから、驚くには当たらない。しかし、ストリームバッファ62によるアブローチのトラフィック率は、常時プリフェッチよりもミスの際のプリフェッチ又はタグ付プリフェッチのそれに遙かに近いに相違ない。

【0054】

【表3】

	fetch	ccom	yacc	net	gxr	liver	linpack
4KB instr. cache, direct-mapped, 16B lines, 1-instr prefetch latency:							
on miss	44.1	42.4	45.2	55.8	47.3	42.8	4
tagged	78.6	74.3	65.7	76.1	89.0	77.2	7
always	82.0	80.3	62.5	81.8	89.5	84.4	8
4KB data cache, direct-mapped, 16B lines, 1-instr prefetch latency:							
on miss	38.2	10.7	14.1	14.5	49.8	75.7	7
tagged	39.7	18.0	21.0	14.8	63.1	83.1	8
always	39.3	37.2	18.6	11.7	63.1	83.8	9

【0055】

【表4】

technique	misses eliminated
for 4KB direct-mapped instruction cache w/16B lines:	
Prefetch on miss (1-instr latency)	46.3
single stream buffer	72.0
quasi-stream buffer (3 comparator)	75.0
tagged prefetch (1-instr latency)	76.8
always prefetch (1-instr latency)	80.1
for 4KB direct-mapped data cache w/16B lines:	
single stream buffer	25.1
prefetch on miss (1-instr latency)	33.1
tagged prefetch (1-instr latency)	40.1
always prefetch (1-instr latency)	42.1
4-way stream buffer	43.1
4-way quasi-stream buffer	47.1

25

【0056】表4はまた、ストリーム バッファ62の性能を、データ参照のためのプリフェッチ技術に替える。茲ではすべての型式の4方向ストリーム バッファ システム80がプリフェッチ戦術を出力実行する。これは主として、プリフェッチ戦術が、たとえそれが必要でない場合にさえも、プリフェッチされた項目を常にキャッシュ内に置くからである。ストリーム バッファ62によるアプローチは、要請されたときのみ項目をキャッシュ20内に動かすので、結果としてプリフェッチされたデータを常にキャッシュ20内に置くよりも汚染が少ない。このことは、データ参照標準の空間的所在が命令参照標準の空間的所在よりも少ないのだから、またプリフェッチされたデータはプリフェッチされた命令よりも遙かに汚染となり易いのだから、データ参照にとって特に重要である。

【0057】ストリーム バッファ62の相対的性能及び理想的なプリフェッチ技術とは無関係に、ストリーム バッファによるアプローチは遙かに実行し易いものである。それはこのアプローチが（逐次参照標準パターンに対するミスの際のプリフェッチ又はタグ付プリフェッチとは異なり）パイプライン化されたメモリ システムの利点を活用できるからである。それはまた、先行のブロックが使用される前にブロックをフェッチすることを開始できるから、プリフェッチされたデータに対する回転待ち時間への要求条件がプリフェッチ技術に較べて低い。最後に、少なくとも命令のストリーム バッファ62に対しては、ストリーム バッファ62が特別に必要とする余分のハードウェアは、タグ付プリフェッチが必要とする余分なタグ記憶部としばしば同程度である。

【0058】小さいミス キャッシュ42（例えば2~8 エントリーの）は、1Kないし8Kバイトの範囲の直接写像されたキャッシュ20に対するデータ キャッシュ衝突ミスの減少に効果的であることが示されている。それらは、キャッシュ中の同じラインに写像する2ないし4ライン間にミスが交互に生じる厳しい衝突を、効果的に除去する。

【0059】ビクティム キャッシュ52は、小さい付属キャッシュ52の内のキャッシュ ミスのビクティムを、ターゲットに代わって救済するミスキャッシュ化の改良である。ビクティム キャッシュ52は、衝突ミスを除去するのにミス キャッシュ42よりもずっと効果的である。ビクティム キャッシュ52は、ラインのサイズが増大し、衝突ミスの百分率が増加するのに伴い一層有益である。衝突ミスの百分率が増加するのに伴い、ビクティム キャッシュ52により除去可能なこれらのミスの百分比も増加し、その結果は、ビクティム キャッシュ52の使用により可能な性能の改善に対し更に険しい傾斜となるように、一般的には見受けられる。

【0060】ストリーム バッファ62は、ミスしたキャッシュ ラインの後のキャッシュラインをプリフェッチ

26

する。ストリーム バッファ62は該ラインを、不必要なキャッシュの汚染を避けるために（キャッシュ ミスがもしあれば）キャッシュミスにより要請されるまで記憶しておく。それは容量及び強制的なミスの数を減少するのに特に有用である。それは、前に論じたタグ付プリフェッチ又はミスの際のプリフェッチのようなプリフェッチ技術とは異なり、逐次参照標準に対するパイプライン化されたメモリ システムで使用可能なメモリ帯域幅を活用できる。ストリーム バッファ62はまた、他のプリフェッチ技術（常時プリフェッチさえも）より遙かに前以てデータをプリフェッチするから、さらに長いメモリシステム回転待ち時間を許容できる。ストリーム バッファ62はまた、命令衝突ミスも同じく相対的に逐次性を持つ傾向がある故、命令衝突ミスを補償することもできる。

【0061】多方向ストリーム バッファ システム80は、同時に発生するいくつかのストリームをプリフェッチできるところの一群のストリーム バッファ62である。この検討では、プリフェッチ開始アドレスが LRU (least recently used - 最長時間未使用) 順ですべてのストリーム バッファ62に互に置き換えられる。多重経路ストリーム バッファ62は、アレイ操作(array operations)におけるようないくつかの異なる巨大データ構造へのインターリーブされたアクセス(interleaved accesses)を含むデータ参照に対し有用である。しかし、プリフェッチすることは逐次ライン(sequential lines)だから、（2又は3の）単位幅(unit stride)のアクセスパターンのみが利益を受ける。

【0062】ビクティム キャッシュ52による性能の改善及びストリーム バッファ62による性能の改善は、データ参照標準に対し相対的に直交(orthogonal)するものである。ビクティムキャッシュ52は、参照標準がキャッシュ中の同じラインに写像する2つの所在位置を交互に往き来するときに好適に働く。それはデータをプリフェッチする訳ではなく、フェッチされたデータを使用可能に維持するためにずっとよい仕事をするだけである。しかるに、ストリーム バッファ62は、データをプリフェッチすることにより性能の改善を達成する。それは衝突が時間的に広い間隔を持つのでない限り衝突ミスを除去しない、そしてキャッシュ ミス参照ストリームは多数の逐次アクセスから成るのである。これらは正に、その相対的に小さい容量の故にビクティム キャッシュ52により旨く処理されない衝突ミスなのである。6つのベンチマークの一群に互って平均的に、4エントリーのビクティム キャッシュに当たった4KBの直接写像されたデータ キャッシュ ミスの僅かに2.5%が、ccom, met, yacc, gcc 及びlivermore に対して、4方向ストリーム バッファにも当たるのである。対照的に、linpack はそのデータ アクセス パターンの故に、ビクティム キャッシュに当たったものの50%が4方向スト



リーム バッファにも当たるのである。しかし、linpack のキャッシュ ミスの僅かに4%がビクティム キャッシュに当たる。それは6つのベンチマークのうちでビクティム キャッシュすることからもたらす利益が最小であり、従ってこのことはまだストリーム バッファとビクティム キャッシュすることとの間の有意な量の重複ではない。

【0063】図20は、ベース システム10に4エントリー データビクティム キャッシュ52と命令ストリーム バッファ62と4方向ストリーム バッファ62部分システムとを付加したものの性能を示すグラフで、該システムは図19A 及び図19B にシステム100 とされているものである。(該ベース システムは、24サイクル ミス ペナルティを持つオンチップ 4KB命令及び 4KBデータ キャッシュ、並びに 128バイト ライン及び320 サイクル ミス ペナルティを持つ3段階第2レベル1MBキャッシュへの16バイト ラインを持つ。) 図20の下の実線はビクティム キャッシュ又はバッファを持たない元のベース システムの性能を表し、上の実線はバッファ及びビクティム キャッシュを持つ場合の性能を表している。これらの技術の組合せは第1レベルのミス レートを、これらの特徴を持たないベースライン システム10のその半分未満に減少させ、結果として6つのベンチマークに亘る平均で 143%のシステム性能の改善がもたらされている。これらの結果は、僅かの量のハードウェアの付加によりキャッシュ ミス レートを劇的に減少させ、システム性能を改善したことを示している。

#### 【図面の簡単な説明】

【図1】図1は、本発明に係わるようなベースラインシステムの概略ブロック図である。  
 【図2】図2は、本発明に係わらない図1のシステムの性能を示すグラフである。  
 【図3】図3は、本発明に係わらない図1のシステムの性能の別の面を示すグラフである。  
 【図4】図4は、本発明の実施例に係わる図1に示すシステムの一部分の概略ブロック図である。  
 【図5】図5は、図4の部分に含まれるシステムの性能のある1つの面を示すグラフである。  
 【図6】図6も、図4の部分に含まれるシステムの性能のまた別の面を示すグラフである。  
 【図7】図7は、本発明のもう1つの実施例に係わる図4の部分に対応するシステムの一部分の概略ブロック図である。  
 【図8】図8は、図7の部分に含まれるシステムの性能のある1つの面を示すグラフである。  
 【図9】図9も、図7の部分に含まれるシステムの性能のある1つの面を示すグラフである。  
 【図10】図10も、図7の部分に含まれるシステムの性能のある1つの面を示すグラフである。  
 【図11】図11も、図7の部分に含まれるシステムの性

能のある1つの面を示すグラフである。

【図12】図12は、本発明に係わらない図1のシステムの性能のもう1つの別の面を示すグラフである。

【図13】図13は、本発明の更にもう1つの実施例に係わる図4の部分に対応するシステムの一部分の概略ブロック図である。

【図14】図14は、図13の部分に含まれるシステムの性能のある1つの面を示すグラフである。

【図15】図15も、図13の部分に含まれるシステムの性能のまた別の面を示すグラフである。

【図16】図16は、本発明の更にもう1つの実施例に係わる図13の部分に対応するシステムの一部分の概略ブロック図である。

【図17】図17は、図16の部分に含まれるシステムの性能のある1つの面を示すグラフである。

【図18】図18も、図16の部分に含まれるシステムの性能のまた別の面を示すグラフである。

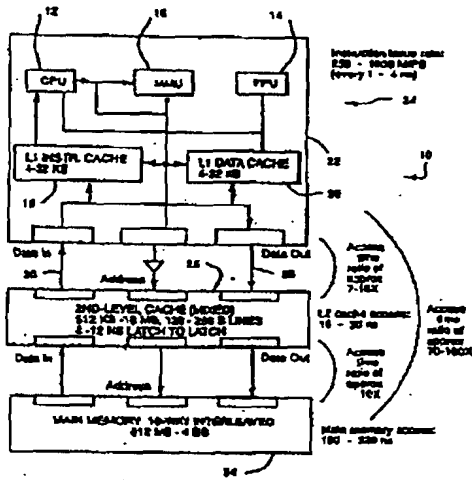
【図19】図19は、図19A と図19B とを左右に並べたものであり、図19A と図19B とはそれぞれ、本発明の更にもう1つの実施例を示す図7、図13及び図16のシステム部分の組合せに一般的に対応するシステムの一部分の概略ブロック図の左半分と右半分とを示す図である。

【図20】図20は、図19A と図19B のシステムの性能のある1つの面を示すグラフである。

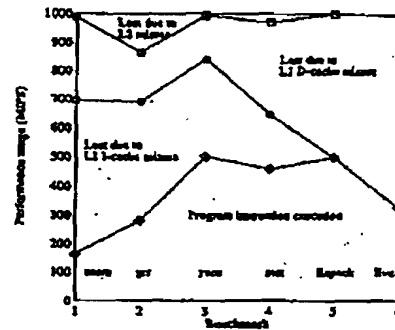
#### 【符号の説明】

- 10 メモリ システム (ベースライン システム)
- 12 CPU (中央処理ユニット)
- 14 フローティング ポイント ユニット (FPU)
- 16 メモリ マネージメント ユニット (MMU)
- 18, 20 データ キャッシュ (第1レベル)
- 22 チップ (プロセッサ チップ)
- 24 セントラル プロセッサ (中央処理装置)
- 26 第2レベル キャッシュ
- 28 ラッチ
- 34 主メモリ
- 40 ミス キャッシュ システム
- 42 ミス キャッシュ
- 44 キャッシュ ライン
- 50 キャッシュ システム
- 52 完全付属ミス キャッシュ (ビクティム キャッシュ)
- 56 ライン
- 60 単一ストリーム バッファ システム
- 62 ストリーム バッファ
- 64 エントリー
- 66 タグ
- 70 データ ライン
- 72 比較器
- 80 多方向ストリーム バッファ システム

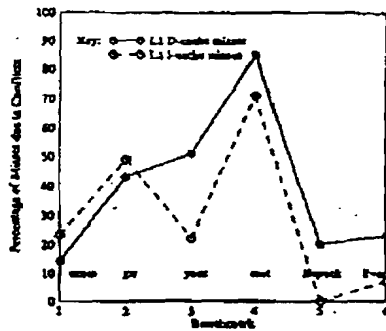
【図1】



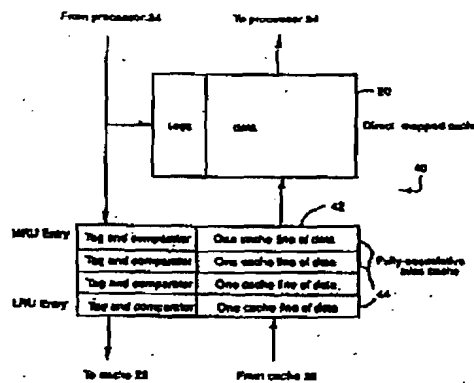
【図2】



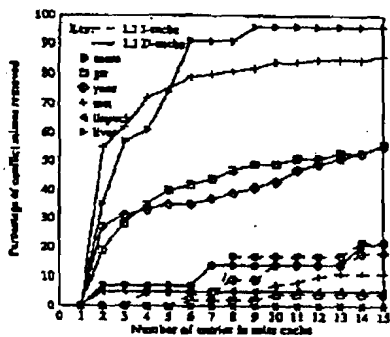
【図3】



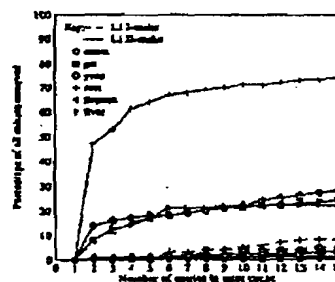
【図4】



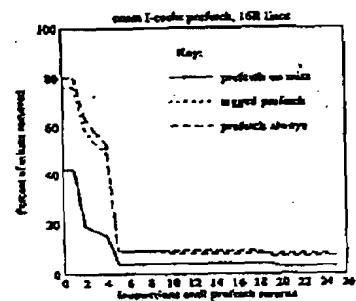
【図5】



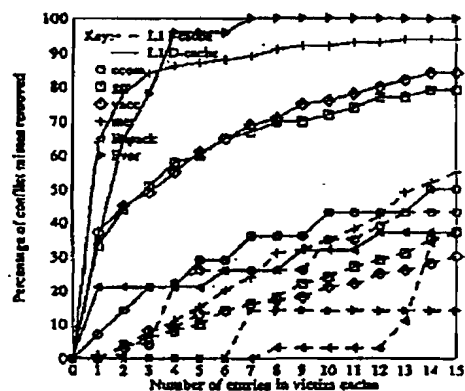
【図6】



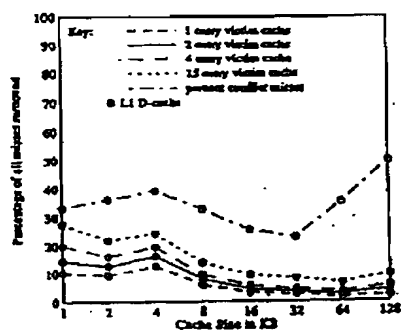
【図12】



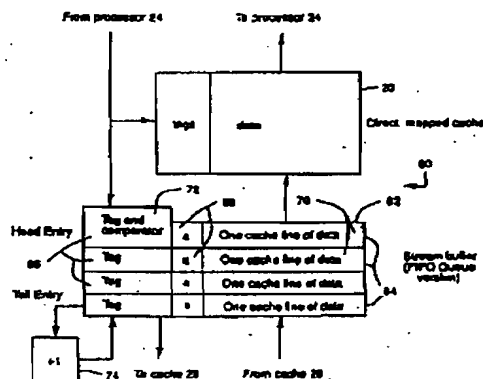
【图 8】



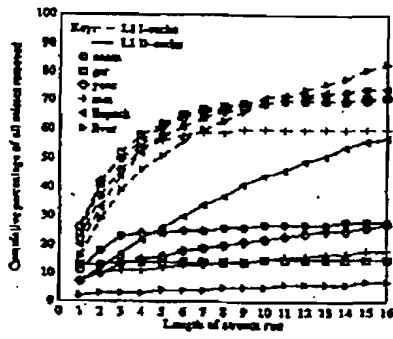
【圖 10】



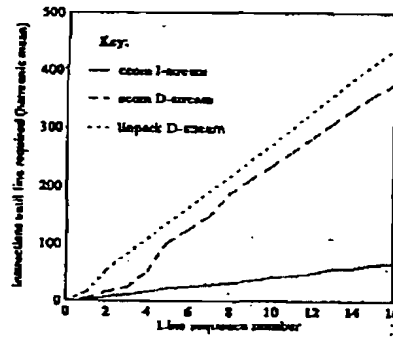
【圖 13】



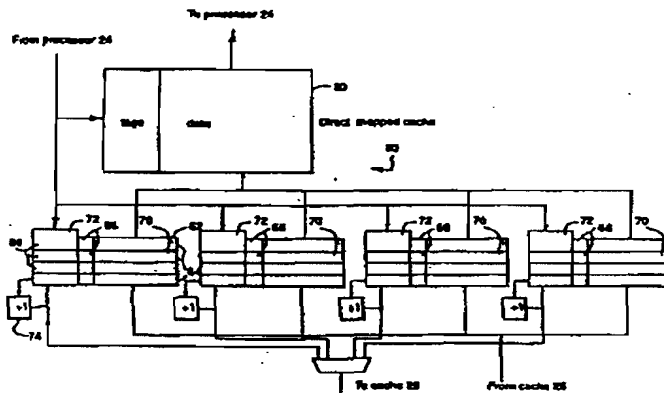
【図14】



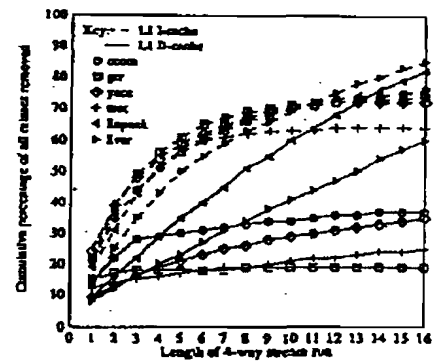
【図15】



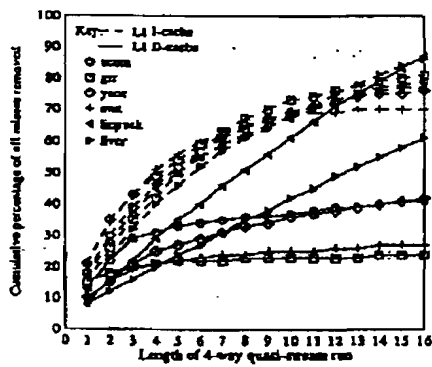
【図16】



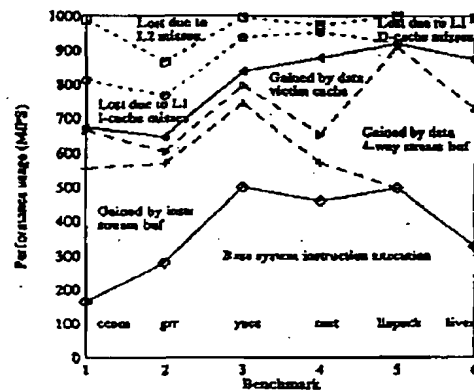
【図17】



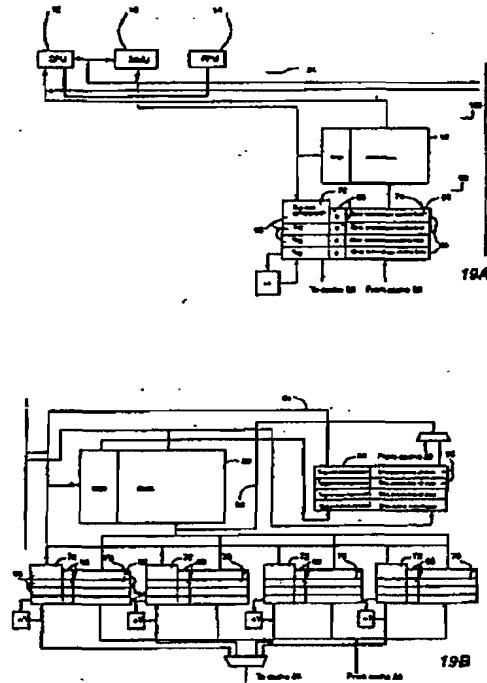
【図18】



【図20】



【図19】



フロントページの続き

(72)発明者 アラン ユーステース  
 アメリカ合衆国 カリフォルニア州  
 94302 パロ アルト ビー・オー・ボツ  
 クス 1605

